

Injecting Abstract Interpretations into Linear Cost Models

David Cachera

ENS Cachan (Bretagne)/IRISA
Campus de Beaulieu
F-35042 Rennes, France
david.cachera@irisa.fr

Arnaud Jobin

Université Rennes 1/IRISA
arnaud.jobin@irisa.fr

We present a semantics based framework for analysing the quantitative behaviour of programs with regard to resource usage. We start from an operational semantics equipped with costs. The dioid structure of the set of costs allows for defining the quantitative semantics as a linear operator. We then present an abstraction technique inspired from abstract interpretation in order to effectively compute global cost information from the program. Abstraction has to take two distinct notions of order into account: the order on costs and the order on states. We show that our abstraction technique provides a correct approximation of the concrete cost computations.

1 Introduction

Static analyses are used to ensure qualitative properties on programs, such as non-reachability of a given set of forbidden states. The abstract interpretation theory encompasses many existing static analyses and allows for systematically designing a variety of new ones by defining abstract semantic domains and transfer functions adapted to the problem under consideration. The main idea of abstract interpretation is to replace concrete semantic computations (often untractable or even uncomputable) by abstract ones which are guaranteed to terminate, hopefully in reasonable time.

In this paper, we are interested in analysing quantitative properties of programs pertaining to the use of resources (time, memory, ...). The computation of quantitative properties of program behaviours suffers from the same drawbacks as their qualitative counterparts, and thus needs adequate abstraction methods. The field of quantitative software analysis has mainly concentrated on the analysis of probabilistic properties, and the various corresponding models have developed their own abstraction techniques. Modelling non functional, but yet non probabilistic behaviour of programs has received less attention.

Our starting point is an operational model of program execution where the cost of each computational step is made explicit. We take as starting point a standard small-step operational semantics expressed as a transition relation between states extended with *costs* associated to each transition. The set of costs is given a dioid (or idempotent semiring) structure with two operators: a “product” operator that combines the costs along an execution path, and a “sum” operator that combines costs coming from different paths. This allows for recasting the operational semantics into a framework of linear operators over a vectorial structure, namely the moduloid of vectors of costs indexed over the set of states.

Seeing the semantics of a program as a linear operator allows to benefit from the nice algebraic properties of these operators. In particular, we are able to easily define two notions of cost for a

whole program execution: a global cost from input to final states, meaningful only if the program terminates, and a more interesting notion of long-run cost, that corresponds to the maximum average of costs accumulated along a cycle of the program semantics and provides an over-approximation of the average cost per transition of long traces. This latter notion is particularly interesting for the analysis of programs with cyclic behaviour (such as reactive systems) in which the asymptotic average cost along cycles, rather than the global cost of the entire execution, is of interest.

Usual abstract interpretations are defined using Galois connections on partially ordered structures, generally assuming the existence of a complete lattice structure for concrete and abstract semantic domains. In our model, we already have a notion of partial order, that is the order on costs induced by the summation operator of the dioid. This order is easily extended pointwise to vectors indexed over states. If we do not assume any additional lattice order on states, we are able to define a simple notion of partition based abstraction. This abstraction technique has been developed in [18, 7], and is suitable for simple analyses that consist in “forgetting” information when going from concrete to abstract states. If we want to use more elaborate abstractions, and in particular reuse the classical abstractions of standard abstract interpretation theory, we have to find an abstraction technique that copes with two distinct notions of order: the dioid order on costs, and the lattice order on states. The present paper addresses more specifically that question.

This paper is structured as follows. Section 2 defines the quantitative operational semantics of a program as a linear operator, and gives the precise definition of cost dioid. Section 3 defines the notions of global and long-run cost that can be extracted from the operational semantics. Section 4 gives the general definition of Galois connection that is used in abstract interpretation theory, and shows its relation with the notion of residuation that is used in our dioid context. Section 5 recalls the main results of partition based abstractions, and shows the limitations of this technique. Section 6 shows how abstractions can be designed that respect both the dioid order of costs and the lattice structure of states. Section 7 gives related work and concluding remarks.

2 Linear operator semantics

Transitions of the semantics are supplied with *quantities* (or *costs*) depending on the accessed states. We consider as semantic model a countable set of states Σ , and define a program as a transition system $P = \langle \Sigma, \rightarrow, I, F \rangle$, where I is a set of initial states and F a set of final states, without referring to any particular syntax. The quantitative operational semantics of P is defined by the transition relation $\rightarrow \subseteq \Sigma \times \Sigma \rightarrow Q$ where a transition $\sigma \rightarrow^q \sigma'$ denotes a transition from state σ to state σ' at cost q . The cost q is function of σ and σ' , and the structure of the set Q of costs will be made precise in the next subsection.

The trace semantics of P is defined as follows.

$$\llbracket P \rrbracket_{tr} = \{ \sigma_0 \rightarrow^{q_0} \dots \sigma_{n-1} \rightarrow^{q_{n-1}} \sigma_n \mid \sigma_0 \in I, \sigma_i \rightarrow^{q_i} \sigma_{i+1} \}$$

In the remainder of this section, we explain in more details the structure we chose for sets of costs, namely cost dioids, before showing how the quantitative operational semantics can be seen as a linear operator over vectorial structures constructed from cost dioids.

2.1 Cost dioid

A transition $\sigma \rightarrow^q \sigma'$ of the quantitative operational semantics states that a direct (one-step) transition from σ to σ' costs q . These unitary transitions can be combined into big-step transitions, using two operators: \otimes for accumulating costs and \oplus to get a “maximum” of different costs. Costs can be defined in more general ways (for instance, one could use a more general algebra of costs as in [1]) but the present definition of costs dioids covers a number of different costs and has interesting computational properties, since it can be used within a linear operator semantic framework, as presented in the next subsection.

The operator \otimes on Q defines the global cost of a sequence of transitions, $\sigma \rightarrow^{q_1} \dots \rightarrow^{q_n} \sigma'$ simply as $q = q_1 \otimes \dots \otimes q_n$. This is written $\sigma \xrightarrow{p, q} \sigma'$ where p is a sequence of states that has σ (resp. σ') as first (resp. last) state.

There may be several ways to reach a state σ' from a state σ , due to the presence of loops and non-determinism in the semantics. Let the set of possible paths be $\Pi_{\sigma, \sigma'} = \{p \mid \sigma \xrightarrow{p, q_p} \sigma'\}$. The global cost between σ and σ' will be defined, using the operator \oplus on Q , to be $q = \bigoplus_{p \in \Pi_{\sigma, \sigma'}} q_p$. Formally, the two operators have to fulfill the conditions of a (commutative) dioid.

Definition 1 A commutative dioid is a structure (Q, \oplus, \otimes) such that

1. Operator \otimes is associative, commutative and has a neutral element e . Quantity e represents a transition that costs nothing.
2. Operator \oplus is associative, commutative and has \perp as neutral element. Quantity \perp represents the impossibility of a transition.
3. \otimes is distributive over \oplus , and \perp is absorbing element for \otimes ($\forall x. x \otimes \perp = \perp \otimes x = \perp$).
4. The preorder defined by \oplus ($a \leq b \Leftrightarrow \exists c : a \oplus c = b$) is an order relation (i.e. it satisfies $a \leq b$ and $b \leq a \Rightarrow a = b$).

A classical result of dioid theory [12]. states that \oplus and \otimes preserve the order \leq , i.e., for all $a, b, c \in Q$ with $a \leq b$, $a \otimes c \leq b \otimes c$ and $a \oplus c \leq b \oplus c$.

By nature, a dioid cannot be a ring, since there is an inherent contradiction between the fact that \oplus induces an order relation and the fact that every element has an inverse for \oplus .

If several paths go from some state σ to a state σ' at the same cost q , we will require that the global cost is also q , i.e. we work with *idempotent dioids*: $q \oplus q = q$ for all q in Q . Note that in an idempotent dioid $a \leq b \Leftrightarrow a \oplus b = b$.

The fact that sets of states may be infinite, together with the use of residuation theory in Section 4 impose our structure to contain the addition of any set of costs¹.

Definition 2 An idempotent dioid is complete if it is closed with respect to infinite sums, and the distributivity law holds also for an infinite number of summands: for any set $X \subseteq Q$, the infinite sum $\bigoplus_{x \in X} x$ exists in the dioid and for all $a \in Q$, $a \otimes (\bigoplus_{x \in X} x) = \bigoplus_{x \in X} (a \otimes x)$.

A complete dioid is naturally equipped with a top element, that we shall write \top , which is the sum of all its elements. We recall that a complete dioid is always a complete lattice, thus equipped with a meet operator \wedge [3]. The notion of long-run cost we will define in Section 3 relies on the computation of an average cost along the transitions of a cycle. This requires the existence of a n th root function.

¹This way, we define a complete sup-semilattice over Q .

	carrier set	\oplus	\otimes	$\sqrt[n]{q}$
Double-idempotent	$\mathbb{Q} \cup \{+\infty, -\infty\}$	min	max	q
	$\mathbb{R} \cup \{+\infty, -\infty\}$	max	min	q
	$\mathcal{P}(S)$	\cap	\cup	q
	$\mathcal{P}(S)$	\cup	\cap	q
Cancellative	$\mathbb{R}_+^m \cup \{+\infty\}$	min	+	$\frac{q}{n}$
Selective	$\mathbb{R}_+ \cup \{+\infty\}$	max	\times	$q^{\frac{1}{n}}$
	$\mathbb{Q} \cup \{+\infty, -\infty\}$	max	+	$\frac{q}{n}$
	$\mathbb{R} \cup \{+\infty, -\infty\}$	min	+	$\frac{q}{n}$

Table 1: Some examples of cost dioids

Definition 3 A dioid (Q, \oplus, \otimes) is equipped with a n th root function if for all q in Q , equation $X^n = q$ has a unique solution in Q , denoted by $\sqrt[n]{q}$.

A sequence containing n transitions, each costing, on average, $\sqrt[n]{q}$, will thus cost q . Some examples of n th root can be found in Table 1. To be able to easily deal with the n th root, we make the assumption that the n th power is \oplus -lower-semicontinuous (\oplus -lsc for short): for all $X \subseteq Q$, $(\bigoplus_{x \in X} x)^n = \bigoplus_{x \in X} x^n$. This assumption and its consequences will be very useful for the theorems relating long-run cost and trace semantics in Section 3. Note that this equality remains true for finite X (in that case the n th power is said a \oplus -morphism).

The following definition summarizes the required conditions for our structure.

Definition 4 (Cost dioid) A cost dioid is a complete and idempotent commutative dioid, equipped with an n th root operation, where the n th power is \oplus -lsc.

Although the definition of cost dioids may seem rather restrictive, we have shown in [7] that many classes of dioids found in the literature are indeed cost dioids. The table displayed on Table 1 gives a non exhaustive example list of cost dioids. The taxonomy is borrowed from [3]: a dioid is *selective*² if for all a, b , $a \oplus b$ is either a or b , *double-idempotent* if both \oplus and \otimes are idempotent, and *cancellative* if for all a, b, c , $a \otimes b = a \otimes c$ and $a \neq \perp$ implies $b = c$.

The most common examples of cost dioids are $(\overline{\mathbb{R}}, \max, +)$ and $(\overline{\mathbb{R}}, \min, +)$, where $\overline{\mathbb{R}}$ stands for $\mathbb{R} \cup \{-\infty, +\infty\}$. The induced orders are, respectively, the orders \leq and \geq over real numbers, extended to $\overline{\mathbb{R}}$ in the usual way. These dioids are at the basis of discrete event systems theory, from which we borrow the notion of long-run cost in Section 3.

2.2 Semantics as linear operators over dioids

Thanks to the multiplication and addition operators of the cost dioid, the set of one-step transitions can be equivalently represented by a *transition matrix* $M \in \mathcal{M}_{\Sigma \times \Sigma}(Q)$ with

$$M_{\sigma, \sigma'} = \begin{cases} q & \text{if } \sigma \rightarrow^q \sigma' \\ \perp & \text{otherwise} \end{cases}$$

Here, $\mathcal{M}_{\Sigma \times \Sigma}(Q)$ stands for the set of matrices with rows and columns indexed over Σ , and values in Q . In the following, a program $P = \langle \Sigma, \rightarrow, I, F \rangle$ will be equivalently denoted as $P = \langle \Sigma, M, I, F \rangle$ where M is the matrix associated to \rightarrow .

²The order induced by a selective dioid is total.

The set $\mathcal{M}_{\Sigma \times \Sigma}(Q)$ is naturally equipped with two operators \oplus and \otimes in the classical way: operator \oplus is extended pointwise, and operator \otimes corresponds to the matrix product (note that the iterate M^n embed the costs for paths of length n). Recall that the dioid is complete, ensuring existence of the sum for each coefficient of the product matrix. The resulting structure is also an idempotent and complete dioid. The order induced by \oplus corresponds to the pointwise extension of the order over Q : $M \leq M' \Leftrightarrow \forall i, j. M_{i,j} \leq M'_{i,j}$. A transition matrix may also be seen as a linear operator on the moduloid $Q(\Sigma)$, which is the analogue of a vector space using a dioid instead of a field for external multiplication.

If E is an idempotent dioid, then for any moduloid V over E the addition operator \oplus defined pointwise is also idempotent, and thus defines a canonical order on V . As for vector spaces, if n is a given integer, E^n , set of vectors with n components in E , is a moduloid. More generally, a vector $u \in E(\Sigma)$, with $|\Sigma| = n$ can be seen as a function $\delta_u : [1, n] \rightarrow E$. Since Q is complete, we can generalize to the infinite countable case: δ_u becomes a mapping from \mathbb{N} to E . The matrix-vector product is defined by: $(Mu)_i = \bigoplus_{j=1}^{+\infty} \delta_M(i, j) \otimes \delta_u(j)$. In this paper, we will keep the matrix notation for the sake of simplicity, even for an infinite set of indices.

3 Global and long-run cost

3.1 Global cost

Let M be the matrix representing the quantitative transitions of a program P . Recall that M^k summarizes the transition costs of all paths of length k . The global cost is then defined by computing the successive iterates of the transition cost matrix until a fixpoint is reached. The transitive closure M^+ thus contains all the transitions costs from any state to any state.

$$M^+ = \bigoplus_{i=1}^{\infty} M^i$$

The global cost of a program is obtained by extracting the input-output cost from this transitive closure.

Definition 5 *The global cost of a program $P = \langle \Sigma, \rightarrow, I, F \rangle$ is defined as*

$$gc(P) = \bigoplus \{M_{i,f}^+ \mid i \in I, f \in F\}$$

Recall that, since we work in a complete semiring, this transitive closure is always defined. The global cost is related to the standard trace semantics by the following result [18].

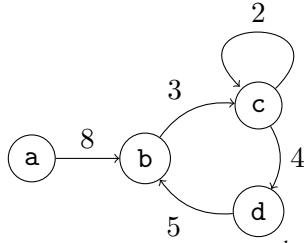
Theorem 1

$$gc(P) = \bigoplus \left\{ \bigotimes_{j=1}^{f-1} q_j \mid \sigma_1 \rightarrow^{q_1} \dots \rightarrow^{q_{f-1}} \sigma_f \in \llbracket P \rrbracket_{tr}, \sigma_f \in F \right\} \quad (1)$$

Unfortunately, if the only information we get is that the global cost is equal to the top element of the dioid, this definition is of little interest. This is the case in particular when the semantics contains cycles of non-null cumulative cost, which frequently arises when matrix M is an abstraction of the semantics, as developed in Section 5. The notion of global cost thus correctly deals with terminating programs over a finite state space, but is inappropriate for reactive systems. For this reason, we rather use the notion of long-run cost.

3.2 Long-run cost

Intuitively, the *long-run cost* of a program represents a maximal average cost over cycles of transitions. The average cost of a finite path is defined as the arithmetical mean (w.r.t. the \otimes operator) of the costs labelling its transitions. In other words, it is the n th root of the global cost of the path, where n is its length. We write $\tilde{q}(p) = \sqrt[|p|]{q(p)}$ for the average cost of path p , where $q(p)$ is the global cost of p , and $|p|$ its length. The “maximum” average cost of all cycles in the graph will be the quantity we are interested in: this quantity will be called *long-run cost*. The following example illustrates these notions on a simple graph.



Average cost of path $abc = (8 + 3)/2 = 5.5$
 Cycle bcd average cost $= (3 + 4 + 5)/3 = 4$
 Cycle $bccdb$ average cost $= 14/4 = 3.5$
 Cycle cc average cost $= 2/1 = 2$
 Long-run cost $= 4$

The diagonal of matrix M^k contains the costs of all cycles of length k . If we add up all the elements on this diagonal, we get the trace of the matrix. This observation gives rise to the following definition.

Definition 6 Let $P = \langle \Sigma, M, I, F \rangle$ a program. Let R be M restricted to the set of states, Σ_I , reachable from I . The long-run cost of program P is defined as

$$\rho(P) = \bigoplus_{k=1}^{|\Sigma_I|} \sqrt[k]{\text{tr } R^k} \quad \text{where} \quad \text{tr } R = \bigoplus_{i=1}^{|\Sigma_I|} R_{i,i}.$$

Note that this definition is valid even for an infinite number of states, since we work with complete dioids. As an example, if we work in the dioid $(\overline{\mathbb{R}}, \max, +)$, $\rho(P)$ may represent the maximal average of time spent per instruction, where the average is computed on any cycle by dividing the total time spent in the cycle by the number of instructions in this cycle. In the case of a finite set of states, the long-run cost is computable, and we note in passing that its definition coincides with the definition of the maximum of eigenvalues of the matrix, in the case of an irreducible matrix in an idempotent semiring [3].

The following proposition [6] establishes in a more formal manner the link between this definition of long-run cost and the cycles of the semantics.

Proposition 1 Let Γ be the set of cycles in \rightarrow^* . Then $\rho(P) = \bigoplus_{c \in \Gamma} \tilde{q}(c)$.

As we aim at giving a characterisation of the asymptotic behaviour of a program, we could have defined the long-run cost as the limit of the average costs of all traces, instead of referring to cycles. The drawback of this approach would be that this definition is not suitable for computation, even if the set of states is finite. It is shown however in [6] that those two notions coincide in a restricted class of cost dioids and when the set of states is finite.

4 Galois connections and residuation

The transition matrix representing a program is in general of infinite dimension, so neither transitive closure nor traces can be computed in finite time. Even if we deal with finitely machine-represented states, the state space is in general too large for ensuring tractable computations.

To overcome this problem, we define an abstract matrix that can be used to approximate the computations of the original matrix. To prove the correctness of this approximation, we re-state the classical abstract interpretation theory [8] in terms of linear operators over moduloids. We first briefly recall a definition of Galois connection that is used in abstract interpretation.

Definition 7 Let (C, \leq_C) and (D, \leq_D) be two partially ordered sets. Two mappings $\alpha : C \mapsto D$ (called abstraction function) and $\gamma : D \mapsto C$ (called concretization function) form a Galois connection (C, α, γ, D) iff:

- $\forall c \in C, \forall d \in D, c \leq_C \gamma(d) \iff \alpha(c) \leq_D d$, or equivalently
- α and γ are monotonic and $\alpha \circ \gamma \leq \text{Id}_D$ and $\text{Id}_C \leq \gamma \circ \alpha$

The classical use of Galois connections considers complete lattices, but their general definition is given on partially ordered sets. A question that naturally arises is that of the existence of an analogous notion relative to vectorial structures. In the case of vector spaces over the field of reals (more precisely, reals between 0 and 1 denoting probabilities), Di Pierro and Wiklicky [11] provide an elegant solution by using the notion of Moore-Penrose pseudo-inverse for bounded linear operators over Hilbert spaces. In our setting, we do not have a field structure, but still benefit from a partial order relation between vectors, namely the order induced by the \oplus operators over vectors in a moduloid. From a general point of view, the α and γ mappings from a Galois connection form a *pair of residuated maps* [14]. Applied to our dioid setting, residuation theory can be restated as follows [3].

Proposition 2 Let E and F be two sets equipped with a complete partial order, f a monotone mapping from E to F . We call subsolution of equation $f(x) = b$ an element y such that $f(y) \leq b$. The following properties are equivalent.

1. For all $b \in F$, there exists a greatest subsolution to the equation $f(x) = b$.
2. $f(\perp_E) = \perp_F$, and f is \oplus -lsc.
3. There exists a monotone mapping $f^\dagger : F \rightarrow E$ which is upper³ semi-continuous such that $f \circ f^\dagger \leq \text{Id}_F$ and $\text{Id}_E \leq f^\dagger \circ f$.

Consequently, f^\dagger is unique. When f satisfies these properties, it is said to be residuated, and f^\dagger is called its residual.

In our framework, the complete orders are the moduloid orders defined pointwise from the cost dioid order. If no additional order on the *set of states* is assumed, there is a straightforward way to define residuable pairs of abstraction and concretization functions on moduloids. This method of abstraction has been developed in [18, 7], and we recall it in the next section to show its limitations. If we start from an already existing abstraction function using a lattice structure on states, we have to cope with two distincts orders: the lattice order on states, and the dioid order on costs. We thus have to define residuated pairs that take both orders into account. This will be developed in Section 6.

³Upper semi-continuity is the analog of lower semi-continuity for the \wedge operator.

5 Partition-based abstraction

We will first consider the simple case where the abstraction is a mapping from concrete to abstract states. This comes down to partitioning the set of concrete states where equivalence classes are defined by abstract states. In this section, Σ will denote a set of *concrete* states and Σ^\sharp a set of *abstract* states, with *no assumption* on the structure of these sets. In particular, they are not supposed to be ordered. An abstraction function α is thus a mapping from Σ to Σ^\sharp . In contrast, we consider a cost dioid Q with its partial order relation.

5.1 Linear operator for abstraction

If we now want to see the abstraction function as a linear abstraction operator between the moduloids constructed on Q with indexes in Σ and Σ^\sharp , respectively, we define the *linear lift* [7] of α as $\alpha^\uparrow \in \mathcal{M}_{\Sigma^\sharp \times \Sigma}(Q)$ by setting⁴

$$\alpha_{\sigma^\sharp, \sigma}^\uparrow = \begin{cases} e & \text{if } \alpha(\sigma) = \sigma^\sharp \\ \perp & \text{otherwise} \end{cases}$$

In order to alleviate notations, \leq will stand for the pointwise order defined on $\mathcal{M}_{\Sigma \times \Sigma}(Q)$ or $\mathcal{M}_{\Sigma^\sharp \times \Sigma^\sharp}(Q)$. The pointwise orders defined on moduloids constructed over a complete dioid are also complete. Moreover, as the abstraction function is linear, it trivially fulfills requirements 2 of Proposition 2, and we get the following result [18] by taking $\gamma^\uparrow = (\alpha^\uparrow)^\dagger$.

Theorem 2 *Let Σ and Σ^\sharp be the domains of concrete and abstract states, α a mapping from Σ to Σ^\sharp , and $\alpha^\uparrow \in \mathcal{M}_{\Sigma^\sharp \times \Sigma}(Q)$ the linear mapping obtained by lifting α . There exists a unique monotonic γ^\uparrow such that*

$$\alpha^\uparrow \circ \gamma^\uparrow \leq \text{Id}_{\Sigma^\sharp \times \Sigma^\sharp} \quad \text{and} \quad \text{Id}_{\Sigma \times \Sigma} \leq \gamma^\uparrow \circ \alpha^\uparrow$$

where $\text{Id}_{\Sigma \times \Sigma}$ (resp. $\text{Id}_{\Sigma^\sharp \times \Sigma^\sharp}$) denotes the identity matrix in $\mathcal{M}_{\Sigma \times \Sigma}(Q)$ (resp. $\mathcal{M}_{\Sigma^\sharp \times \Sigma^\sharp}(Q)$).

The very simple form of abstraction we deal with up to now gives rise to a very simple expression for γ^\uparrow . Indeed, the unique γ^\uparrow matching the requirements of Theorem 2 is the transpose matrix of α^\uparrow .

5.2 Induced abstract semantics

Given a program P over Σ , we want to define an abstract transition system over the abstract domain Σ^\sharp that is “compatible” with P , both from the point of view of its traces and from the costs it leads to compute. The following definition of a correct abstraction ensures that both global and long-run costs of P are correctly over-approximated during the abstraction process.

Definition 8 (Correct abstraction) *Let $P = \langle \Sigma, M, I, F \rangle$ be a transition system where $M \in \mathcal{M}_{\Sigma \times \Sigma}(Q)$ and $P^\sharp = \langle \Sigma^\sharp, M^\sharp, I^\sharp, F^\sharp \rangle$ be a transition system over the abstract domain, with $M^\sharp \in \mathcal{M}_{\Sigma^\sharp \times \Sigma^\sharp}(Q)$. Let α be a mapping from Σ to Σ^\sharp . The triple (P, P^\sharp, α) is a correct abstraction from Σ to Σ^\sharp if the three conditions (1) $\alpha^\uparrow \circ M \leq M^\sharp \circ \alpha^\uparrow$, (2) $\{\alpha(\sigma) \mid \sigma \in I\} \subseteq I^\sharp$ and (3) $\{\alpha(\sigma) \mid \sigma \in F\} \subseteq F^\sharp$ hold.*

⁴Recall that e denotes the neutral element for \otimes .

The classical framework of abstract interpretation gives a way to define a best correct abstraction for a given concrete semantic operator. In the same way, given an abstraction α and a concrete semantics linear operator, we can define an abstract semantics operator that is correct by construction, as expressed by the following proposition [7].

Proposition 3 *Let α be an abstraction from Σ to Σ^\sharp , and $P = \langle \Sigma, M, I, F \rangle$ a transition system over the concrete domain. We set $P^\sharp = \langle \Sigma^\sharp, M^\sharp, I^\sharp, F^\sharp \rangle$ with*

$$M^\sharp = \alpha^\uparrow \circ M \circ \gamma^\uparrow \quad \text{and} \quad I^\sharp = \{\alpha(\sigma) \mid \sigma \in I\} \quad \text{and} \quad F^\sharp = \{\alpha(\sigma) \mid \sigma \in F\}$$

Then (P, P^\sharp, α) is a correct abstraction from Σ to Σ^\sharp . Moreover, given P and α , P^\sharp provides the best possible abstraction in the sense that if $P' = \langle \Sigma^\sharp, M', I', F' \rangle$ is another correct abstraction, then $M^\sharp \leq M'$ and $I^\sharp \subseteq I'$ and $F^\sharp \subseteq F'$.

5.3 Correctness of cost computations

The question that naturally arises is to know how global and long-run costs are transformed by abstraction. Theorems 3 and 4 below state that a correct abstraction gives an over-approximation of the concrete global cost [6] and concrete long-run cost [7], respectively.

Theorem 3 *If (P, P^\sharp, α) is a correct abstraction, then $gc(P) \leq_Q gc(P^\sharp)$.*

Theorem 4 *If (P, P^\sharp, α) is a correct abstraction, then $\rho(P) \leq_Q \rho(P^\sharp)$.*

The proofs of these theorems rely on the fact that the correctness is preserved when the concrete and abstract matrices are iterated simultaneously [6].

5.4 Limitations

Partition based abstraction is well adapted to simple cases where abstraction consists in “forgetting” information when going from the concrete state to the abstract one. Let us take an example to illustrate this fact. The concrete operational semantics of an object oriented bytecode language considers states as tuples $(h, (m, pc, l, s) :: sf)$, where h is the heap of objects, and $(m, pc, l, s) :: sf$ is a call stack consisting of *frames* of the form (m, pc, l, s) where each frame contains a method name m and a program point pc within m , a set of local variables l , and a local operand stack s (see [17] for details on such an example). Depending on the property the analysis wants to establish, a first abstraction could define an abstract state as a simpler tuple (h, m, pc, l, s) , making the analysis context-insensitive. If we want to go further, we might want to abstract the heap h , which is usually a mapping from locations to objects, by an abstract heap mapping any location to the class of the corresponding object in the concrete heap. Both of these abstractions are easily expressed by abstraction functions partitioning the set of concrete states, and thus fit well the framework described above.

In contrast, if we now want to abstract the values of local variables by intervals, as is common in static analysis, we face two problems. The first one is similar to a “state explosion” problem, and the second one is related to the translation of the lattice order of intervals into the moduloid structure over abstract states. Let us explain both concerns in more details. Let n be a natural number. We denote by $Int_n^{[2]}$ the set of intervals with even bounds over $\{-n, \dots, n\}$. The interval abstraction function $\alpha_{Int_n^{[2]}} : \mathcal{P}(\{-n, \dots, n\}) \rightarrow Int_n^{[2]}$ maps a set of natural numbers

$\{i_1, \dots, i_r\}$ to the interval $[m - (m \bmod 2), M + (M \bmod 2)]$ where $m = \min_{k \in \{1, \dots, r\}} i_k$ and $M = \max_{k \in \{1, \dots, r\}} i_k$. If we lift $\alpha_{\text{Int}_n^{[2]}}$ into a linear map as above, we get a linear mapping from a moduloid of dimension 2^{2n+1} to a moduloid of dimension $\frac{n(n+1)+2}{2}$. The corresponding matrix is thus of size $\frac{n(n+1)+2}{2} \times 2^{2n+1}$. One could argue that the subsets of $\{-n, \dots, n\}$ could canonically be represented by a moduloid of dimension $2n+1$, each element contributing for one dimension, and thus reducing the matrix size. For instance, if we fix $n=2$, $\{-2\}$ is represented by $(e, \perp, \perp, \perp, \perp, \perp)^T$, $\{2\}$ by $(\perp, \perp, \perp, \perp, e)^T$, $\{-2, 2\}$ by $(e, \perp, \perp, \perp, e)^T$ etc. Let us now examine the abstract domain of even intervals. The set of even intervals over $\{-2, \dots, 2\}$ is lifted to a moduloid of dimension 7. For instance, $[-2]$ is represented as $(\perp, e, \perp, \perp, \perp, \perp, \perp)^T$, and $[2]$ as $(\perp, \perp, \perp, e, \perp, \perp, \perp)^T$, if we order the intervals by increasing size and increasing lower bound. We thus should set $\alpha_{\text{Int}_2^{[2]}}((e, \perp, \perp, \perp, \perp)^T) = (\perp, e, \perp, \perp, \perp, \perp, \perp)^T$, and $\alpha_{\text{Int}_2^{[2]}}((\perp, \perp, \perp, \perp, e)^T) = (\perp, \perp, \perp, e, \perp, \perp, \perp)^T$. Then $\alpha_{\text{Int}_2^{[2]}}((e, \perp, \perp, \perp, e)^T) = (\perp, \perp, \perp, \perp, \perp, \perp, e)^T$, that is distinct from $\alpha_{\text{Int}_2^{[2]}}((e, \perp, \perp, \perp, \perp)^T) \oplus \alpha_{\text{Int}_2^{[2]}}((\perp, \perp, \perp, \perp, e)^T)$ which equals $(\perp, e, \perp, \perp, e, \perp, \perp)^T$. In conclusion, we are not able to define $\alpha_{\text{Int}_n^{[2]}}$ as a linear operator as expected. The problem here comes from the fact that the structure of the abstract moduloid totally forgets about the lattice structure of intervals. Defining a residuable abstraction operator that respects the lattice structure of abstract states is the main contribution of this paper, and is developed in the next section.

6 Lifting Abstract Interpretations

In Section 5, we have presented a way to lift any abstraction function $\alpha : \Sigma \rightarrow \Sigma^\sharp$ into a linear mapping $\alpha^\uparrow \in \mathcal{M}_{\Sigma^\sharp \times \Sigma}(Q)$, where domains Σ and Σ^\sharp are not supposed to have a particular structure. In order to benefit from the already existing abstractions provided by the classical abstract interpretation theory, we show how to translate them into our model. As abstract interpretation relies on lattices and Galois connections, we will investigate in Section 6.1 how these structures compare and are transposed to moduloids and linear operators. Then, in Section 6.2, we will investigate a new notion of correctness for this construction.

6.1 Abstraction operator

So far, the way we lift an abstraction represents a state σ of Σ by a vector $(\perp, \dots, \perp, e, \perp, \dots, \perp)^T$ where e appears in the σ -place (recall Σ is countable). The set of concrete states Σ is thus represented using the moduloid $\Sigma^\uparrow = (\{\perp, e\}^{|\Sigma|}, \oplus, \otimes)$. Now, if we assume that Σ is a lattice, this lifting unfortunately forgets about the ordered structure of Σ . This is regrettable because Σ^\uparrow naturally has a lattice structure given by the \oplus operator. Thus, a natural issue is to translate Σ and Σ^\sharp into moduloids while preserving their respective lattice orders. This property of morphism between orders will be referred to as the lift-order property in the remainder of this section.

6.1.1 Lifting a Galois connection into a linear mapping

Abstract interpretation often considers Galois connections $B \xrightleftharpoons[\alpha]{\gamma} A$ where B is a powerset⁵ representing the concrete semantic domain, and A is a complete lattice representing the abstract

⁵Powersets are naturally equipped with a particular structure of complete lattice called boolean lattice [9].

domain. In order to lift α into a linear mapping, we will focus on how to lift-order these particular structures. The easy case naturally is the one of boolean lattices.

Lift-ordering boolean lattices. A boolean lattice B is generated by its set of atoms $\mathcal{A}(B)$, corresponding to the singletons in the case of a powerset. Indeed, for each $b \in B$, $b = \vee\{a \in \mathcal{A}(B) \mid a \leq b\}$ [9]. Let us code atoms a as vectors a^\uparrow in $\{\perp, e\}^{|\mathcal{A}(B)|}$ as previously (we note $\bar{a} = a^\uparrow$). Then, coding the other elements will follow from the use of \oplus .

$$\bar{b} = \oplus\{\bar{a} \mid a \leq b\}$$

We denote by \bar{B} the complete moduloid constructed this way from B , where the \oplus operator of \bar{B} matches the \cup operator of B by construction.

Now that we have expressed boolean lattices as moduloids, we are able to easily lift-order the abstraction function of a Galois connection $B_1 \xleftarrow[\alpha]{\gamma} B_2$, where B_1 and B_2 are boolean lattices. By lift-ordering these lattices, we obtain two moduloids $(\bar{B}_1, \oplus_1, \otimes_1)$ and $(\bar{B}_2, \oplus_2, \otimes_2)$. Since \cup_i and \oplus_i coincide, and as α is a union morphism, its linear translation $\bar{\alpha}$ is defined by its values on the basis vectors of \bar{B}_1 , *i.e.* the vectors coding atoms of B_1 .

$$\begin{array}{ccccc} \alpha(\{b_1\}) & \cup_1 & \{b_2\} & = & \alpha(\{b_1\}) & \cup_2 & \alpha(\{b_2\}) \\ \uparrow & & & & \uparrow & & \uparrow \\ \bar{\alpha}(\bar{b}_1) & \oplus_1 & \bar{b}_2) & = & \bar{\alpha}(\bar{b}_1) & \oplus_2 & \bar{\alpha}(\bar{b}_2) \end{array}$$

Lift-ordering complete lattices. In most of the cases, A is not a powerset but a more general complete lattice for which the vectorial translation is not so straightforward. The representation theorem of finite distributive lattices [9] asserts that any such lattice A is isomorphic to a lattice of sets. Thus, A can be seen as a sublattice of a given powerset, which we will denote by $\mathcal{B}(A)$. The previous coding applies to $\mathcal{B}(A)$ and *a fortiori* to A . However, the set of vectors \bar{A} constructed this way no more has a structure of complete moduloid, unlike $\mathcal{B}(A)$. This method provides a solution to the “state explosion” problem presented in Section 5. Nevertheless, our second problem remains unsolved. Indeed, there is still no match between the \oplus operator and \cup , the join operator of the lattice. For instance, $[-2] \cup [2] = [-2, 2]$ and $\overline{[-2]} \oplus \overline{[2]} = (e, \perp, e)^T$ and $\overline{[-2, 2]} = (e, e, e)^T$. This makes it impossible to express $\bar{\alpha}$ as a linear mapping, since for instance $\bar{\alpha}(\overline{\{-2\}} \oplus \overline{\{2\}}) = (e, e, e)^T \neq \bar{\alpha}(\overline{\{-2\}}) \oplus \bar{\alpha}(\overline{\{2\}}) = (e, \perp, e)^T$. We thus have to weaken our requirement: in the following, we choose to lift-order Galois connections into non linear, but still residuable, mappings.

6.1.2 Lifting a Galois connection into a residuable mapping

Since $\overline{\mathcal{B}(A)}$ is a complete boolean lattice, we will decompose $\bar{\alpha}$ into a linear part from \bar{B} to $\overline{\mathcal{B}(A)}$, and a projection from $\overline{\mathcal{B}(A)}$ into its sublattice \bar{A} we are interested in, representing the vector encodings of elements of A . Figure 1(b) illustrates this decomposition.

The linear part of $\bar{\alpha}$, denoted by $\bar{\alpha}_1$ is defined as in the case of a connection between two boolean lattices: $\bar{\alpha}_1$ is defined on the set of atoms of B by $\bar{\alpha}_1(\bar{b}) = \overline{\alpha(b)}$ where b is an atom of B , and then extended to \bar{B} by linearity. As an example, Figure 1(c) shows the abstraction matrix for the abstraction by even intervals, for $n = 2$. Element $\{-1\}$ of the concrete domain is mapped to interval $[-2, 0]$ of the abstract domain. Thus, $\bar{\alpha}_1$ maps atom $\overline{\{-1\}}$ to $\overline{[-2, 0]}$ which is the sum of atoms $\overline{[-2]}$ and $\overline{[0]}$.

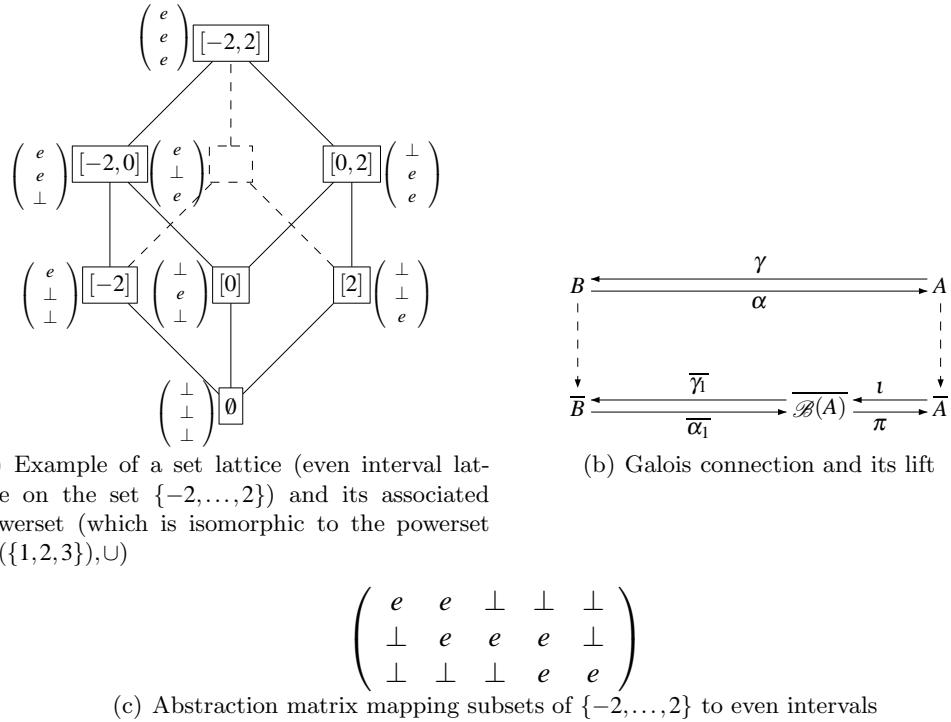


Figure 1: Lifting of Galois connections

This linear mapping is then composed with a projection π in order to yield a vector in \overline{A} corresponding to an element of the (non boolean) lattice A . As we want to keep the lift-order property, for all $x \in \mathcal{B}(A)$, $\pi(x)$ is defined as the smallest element $z \in \overline{A}$ such that $z \geq x^6$. Note that π defined this way is an upper closure operator in $\mathcal{B}(A)$. On our even interval abstraction example, $\overline{\alpha}_1(\overline{\{-2, 2\}}) = (e, \perp, e)^T$ is projected to the top element $(e, e, e)^T$ of the abstract vector lattice.

As $\overline{\alpha}_1$ is a linear mapping between two complete moduloids, by Proposition 2, $\overline{\alpha}_1$ has a residual mapping $\overline{\gamma}_1$, i.e. $\overline{\alpha}_1 \circ \overline{\gamma}_1 \leq \text{Id}_{\overline{\mathcal{B}(A)}}$ and $\overline{\gamma}_1 \circ \overline{\alpha}_1 \geq \text{Id}_{\overline{B}}$. Passing from \overline{A} to $\overline{\mathcal{B}(A)}$ is simply done by a canonical injection ι .

We finally prove the following property, that allows for defining a pseudo-invertible lift of our initial Galois connection (the proof is given in Appendix).

Proposition 4 *Mappings $\pi \circ \overline{\alpha}_1$ and $\overline{\gamma}_1 \circ \iota$ as defined above are such that $\pi \circ \overline{\alpha}_1$ is residuated and $\overline{\gamma}_1 \circ \iota$ is its residual, and thus form a Galois connection between moduloids \overline{B} and \overline{A} seen as lattices.*

By this construction, we are able to translate a Galois connection $B \xleftarrow{\gamma} A$ into a residuable mapping written as the composition of a linear mapping $\overline{\alpha}_1$ and an upper closure operator π . We now check that this new construction of abstraction operators preserves the over-approximation of cost computations.

⁶The completeness property of A and the morphism between the orders on A and on its lifted version \overline{A} ensure the existence of this element.

6.2 Correctness of cost computations

The way of lifting abstraction functions in Section 5.1 produced linear mappings. On the other hand, the lift version of Section 6.1 gives only residuable mappings. Unfortunately, the correctness of cost computations intimately depends on the matricial character of the abstraction function and seems difficult to establish for general residuable mappings. Nonetheless, we can establish the correctness of cost computations in a weaker way, using only the linear part $\overline{\alpha}_1$ of the residuable abstraction $\pi \circ \overline{\alpha}_1$. We thus slightly change the definition of a correct abstraction into a notion of *correct linear abstraction*.

Definition 9 (Correct linear abstraction) *Let \overline{B} and $\overline{\mathcal{B}(A)}$ be two moduloids of respective bases \overline{b} and \overline{ba} . Let Q be a cost dioid. Let $P = \langle B, M, I, F \rangle$ a transition system with $M \in \mathcal{M}_{\overline{b}\overline{b}}(Q)$ and $P^\sharp = \langle \mathcal{B}(A), M^\sharp, I^\sharp, F^\sharp \rangle$ be a transition system over the abstract domain, with $M^\sharp \in \mathcal{M}_{\overline{ba}\overline{a}}(Q)$. Let $\overline{\alpha}_1$ be a linear mapping from \overline{B} to $\overline{\mathcal{B}(A)}$. The triple $(P, P^\sharp, \overline{\alpha}_1)$ is a correct linear abstraction from B to $\mathcal{B}(A)$ if the three conditions (1) $\overline{\alpha}_1 \circ M \leq M^\sharp \circ \overline{\alpha}_1$, (2) $\{\alpha(\sigma) \mid \sigma \in I\} \subseteq I^\sharp$ and (3) $\{\alpha(\sigma) \mid \sigma \in F\} \subseteq F^\sharp$ hold.*

In contrast with Definition 8 where we considered an abstraction function α and stated the correctness using its lifted version $\overline{\alpha}$, we directly consider here the abstraction function as a linear mapping between moduloids⁷. As a consequence, we will prove a notion of correctness that is independent of the way domains are lifted. As far as the global cost is concerned, this makes no difference since Lemma 2 remains true for this notion. However, the correctness proof is more difficult to achieve for the long-run cost, and will require an additional hypothesis on the cost dioid, namely it being selective.

As the notion of long-run cost can be stated without considering initial and final states, in what follows we use the notation of a correct linear abstraction $(M, M^\sharp, \overline{\alpha}_1)$ to refer to the inequality stated in item (1) of Definition 9.

Theorem 5 below states that a correct linear abstraction gives an overapproximation of the global cost, while Theorem 6 states the same result for the long-run cost.

Theorem 5 *If $(M, M^\sharp, \overline{\alpha}_1)$ is a correct linear abstraction, then $gc(M) \leq gc(M^\sharp)$.*

Theorem 6 *Let Q be a selective cost dioid⁸. If $(M, M^\sharp, \overline{\alpha}_1)$ is a correct linear abstraction, then $\rho(M) \leq \rho(M^\sharp)$.*

As mentioned above, proof of Theorem 5 is a direct consequence of Lemma 2. On the contrary, proof of Theorem 6 requires four lemmas, whose proofs are given in Appendix.

Lemma 1 *Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction and $(\sigma^\sharp, \sigma) \in \overline{ba} \times \overline{b}$. Then, we have:*

$$\bigoplus_{\{c \in \overline{b} \mid \sigma^\sharp \leq \overline{\alpha}_1(c)\}} M_{c\sigma} \leq \bigoplus_{\{a \in \overline{ba} \mid a \leq \overline{\alpha}_1(\sigma)\}} M_{\sigma^\sharp a}^\sharp. \quad (2)$$

Lemma 1 is quite straightforward. Its proof consists in developing each member of the correct linear abstraction inequality. Lemma 2 is not specific to this section and was borrowed from the proof of Theorem 4.

Lemma 2 *Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction. Then, for all $k \geq 1$, $(M^k, (M^\sharp)^k, \overline{\alpha}_1)$ is a correct linear abstraction.*

⁷For instance, this can be achieved by applying techniques of Section 6.1 on a Galois connection $B \xleftarrow{\gamma} \overline{A}$

⁸Recall that a dioid is *selective* if for all a, b , $a \oplus b$ is either a or b

Lemma 3 is the core of Theorem 6. It establishes that every cycle of length k of the concrete graph represented by M has a corresponding abstract path of the same length k and of higher cost. As mentioned above, we will assume that the cost dioid is selective.

Lemma 3 *Let us assume a selective cost dioid Q . Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction. Then, for all σ and $k \geq 1$, such that $M_{\sigma\sigma}^k \neq \perp$ and for all $\sigma_i^\sharp \in \overline{ba}$ appearing in the vector decomposition of $\overline{\alpha}_1(\sigma)$ on the basis \overline{ba} , there exists $\sigma_j^\sharp \leq \overline{\alpha}_1(\sigma)$ such that $M_{\sigma_i^\sharp \sigma_j^\sharp}^{\sharp k} \geq M_{\sigma\sigma}^k$.*

Finally, Lemma 4 states that for every cycle of length k of the concrete graph M there exists a cycle of potentially higher length kr in the abstract graph M^\sharp and of higher average cost.

Lemma 4 *Let Q be a selective cost dioid. Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction. Let $\sigma \in \overline{b}$ and $k \geq 1$ such that $M_{\sigma\sigma}^k \neq \perp$. We note $\overline{\alpha}_1(\sigma) = \sigma_1^\sharp \oplus \dots \oplus \sigma_s^\sharp$. Then, there exist $1 \leq j, r \leq s$ such that:*

$$\sqrt[k]{(M^k)_{\sigma\sigma}} \leq \sqrt[kr]{(M^{\sharp kr})_{\sigma_j^\sharp \sigma_j^\sharp}}.$$

7 Discussion and related work

We have defined a quantitative counterpart of abstract interpretation starting from an operational semantics where transitions are labelled with costs of computations. The dioid structure of the set of costs allows for defining concrete and abstract semantics as linear operators between moduloids. We have presented two abstraction techniques for relating concrete and abstract semantics. The first one defines an abstraction function as a linear operator. It is usable for simple cases of abstractions, but suffers from a state explosion problem and is not suited for reusing standard abstract domains provided by the abstract interpretation literature. The second technique decomposes abstraction into a linear operator and a projection operator, and establishes a link between our framework and standard Galois connections. We have shown that both techniques provide an over-approximation of concrete cost computations.

This article follows [18, 7], where the first abstraction technique was presented. It broadens our view of quantitative static analysis by allowing a reuse of classical abstract domains used in qualitative static analyses.

The present work is inspired by the quantitative abstract interpretation framework developed by Di Pierro and Wiklicky [11]. We have followed their approach in modeling programs as linear operators over a vector space, with the notable technical difference that their operators act over a semiring of probabilities whereas ours work with idempotent dioids. In Di Pierro and Wiklicky's work, the relation with abstract interpretation is justified by the use of the pseudo-inverse of a linear operator, similar to a Galois connection mechanism, enforcing the soundness of abstractions. Our approach can be seen as intermediate between their and classical abstract interpretation: on one hand, we use residuation theory in order to get a pseudo-inverse for linear abstraction functions; on the other hand, we benefit from the partially ordered structure of dioids to give guarantees of soundness under the assumption $\alpha \circ M \leq_D M^\sharp \circ \alpha$, which is a classical requirement in abstract interpretation. Another approach for probabilistic abstract interpretation has been followed by Monniaux [15] for the analysis of imperative programs containing random operators, where the semantics of a program is seen as a mapping between probability distributions. Note however that none of the dioid approach and the probabilistic semiring approach can generalise the other one, since there is an inherent contradiction between being a ring and a dioid. Reconciling both frameworks would require the definition of a more complex

mathematical structure equipped with all operators, with the difficulty of keeping all the nice properties of the initial models.

Several other works make use of idempotent semirings for describing quantitative aspects of computations, namely under the form of constraint semirings [4], particularly under the name of *soft constraints*. These have been used in the field of Quality of Service [10, 16], in particular with systems modelled by graph rewriting mechanisms [13]. In all these approaches, the \oplus and \otimes operators of the constraint semiring are used for combining constraints. Among these works, two similar approaches deserve a particular attention, since they deal with abstraction mechanisms. Aziz [2] makes use of semirings in a mobile process calculus derived from the π -calculus, in order to model the cost of communicating actions. He also defines a static analysis framework, by abstracting “concrete” semirings into abstract semirings of reduced cardinality, and defining abstract semiring operators accordingly. Bistarelli *et al.* [5] define an abstract interpretation based framework for abstracting soft constraint satisfaction problems (SCSPs). As in Aziz’s approach, they get an abstract SCSP by just changing the associated semiring, leaving unchanged the remainder of the structure. Concrete and abstract semirings are related by means of a Galois insertion, which provides correctness results. A major difference between these approaches and ours is that they abstract the semiring and leave the system itself unchanged, while we abstract the structures of states and keep the same dioid.

This paper tackles the problem of the linear operator approach for modelling quantitative semantics. Even if we managed to get residuated pairs for translating Galois connection into a linear model, the correctness of cost computations for a lifted Galois connection is defined only for its linear part, thus forgetting about the final projection. One could argue that this correctness is not adequate, since it does not deal with the final abstract semantics but with an intermediate one. Recall however that we aim at computing an over-approximation of the *concrete* long-run cost. Thus, the fact that the “exact” abstract semantics is obtained by a subsequent projection does not really matter here.

An interesting avenue for further work would be to relax the correctness criterion so that the abstract estimate is “close” to (but not necessarily greater than) the exact quantity. For certain quantitative measures, a notion of “closeness” might be of interest, as opposed to the qualitative case where static analyses must err on the safe side.

References

- [1] David Aspinall, Lennart Beringer, Martin Hofmann, Hans-Wolfgang Loidl & Alberto Momigliano (2007): *A program logic for resources*. *Theor. Comput. Sci.* 389(3), pp. 411–445.
- [2] Benjamin Aziz (2006): *A Semiring-based Quantitative Analysis of Mobile Systems*. *Electronic Notes in Theoretical Computer Science* 157(1), pp. 3–21.
- [3] François Baccelli, Guy Cohen, Geert Jan Olsder & Jean-Pierre Quadrat (1992): *Synchronization and Linearity*. Wiley.
- [4] Stefano Bistarelli (2004): *Semirings for Soft Constraint Solving and Programming*, Lecture Notes in Computer Science 2962. SpringerVerlag.
- [5] Stefano Bistarelli, Philippe Codognet & Francesca Rossi (2002): *Abstracting soft constraints: framework, properties, examples*. *Artif. Intell.* 139(2), pp. 175–211.
- [6] David Cachera, Thomas Jensen, Arnaud Jobin & Pascal Sotin (2007): *Long-Run Cost Analysis by Approximation of Linear Operators over Dioids*. Research Report 6338, INRIA. Available at <https://hal.inria.fr/inria-00182338>.

- [7] David Cachera, Thomas Jensen, Arnaud Jobin & Pascal Sotin (2008): *Long-Run Cost Analysis by Approximation of Linear Operators over Diods*. In: *Proc. of the 12th International Conference on Algebraic Methodology and Software Technology*, number 5140 in Lecture Notes in Computer Science, Springer Verlag, Urbana, Illinois, USA, pp. 122–138.
- [8] Patrick Cousot & Radhia Cousot (1977): *Abstract Interpretation: A Unified Lattice Model for Static Analysis of Programs by Construction or Approximation of Fixpoints*. In: *4th Symposium on Principles of Programming Languages (POPL'77)*, pp. 238–252.
- [9] B. A. Davey & H. A. Priestley (1990): *Introduction to Lattices and Order*. Cambridge University Press.
- [10] Rocco De Nicola, Gianluigi Ferrari, Ugo Montanari, Rosario Pugliese & Emilio Tuosto (2005): *A Basic Calculus for Modelling Service Level Agreements*. In: *International Conference on Coordination Models and Languages, Lecture Notes in Computer Science 3454*, Springer Verlag, Namur (Belgium), pp. 33 – 48.
- [11] Alessandra Di Pierro & Herbert Wiklicky (2000): *Concurrent Constraint Programming: Towards Probabilistic Abstract Interpretation*. In: *Principles and Practice of Declarative Programming*, pp. 127–138.
- [12] Michel Gondran & Michel Minoux (2008): *Graphs, Diods and semirings, New Models and Algorithms*. Springer.
- [13] Dan Hirsch & Emilio Tuosto (2005): *SHReQ: Coordinating Application Level QoS*. In: *SEFM '05: Proceedings of the Third IEEE International Conference on Software Engineering and Formal Methods*, IEEE Computer Society, Washington, DC, USA, pp. 425–434.
- [14] A Melton, D A Schmidt & G E Strecker (1986): *Galois connections and computer science applications*. In: *Proceedings of a tutorial and workshop on Category theory and computer programming*, Springer-Verlag New York, Inc., New York, NY, USA, pp. 299–312.
- [15] David Monniaux (2000): *Abstract interpretation of probabilistic semantics*. In: *Static Analysis Symposium (SAS '00)*, number 1824 in Lecture Notes in Computer Science, Springer Verlag, pp. 322–339.
- [16] Francesco Santini (2008): *Managing quality of service with soft constraints*. In: *AAAI'08: Proceedings of the 23rd national conference on Artificial intelligence*, AAAI Press, pp. 1869–1870.
- [17] Igor Siveroni (2004): *Operational Semantics of the Java Card Virtual Machine*. *J. Logic and Automated Reasoning*.
- [18] Pascal Sotin, David Cachera & Thomas Jensen (2006): *Quantitative Static Analysis over Semirings: Analysing Cache Behaviour for Java Card*. In: Alessandra Di Pierro & Herbert Wiklicky, editors: *QAPL06, Quantitative Aspects of Programming Languages*.

A Proof of Proposition 4

Proposition 4 *Mappings $\pi \circ \overline{\alpha_1}$ and $\overline{\gamma_1} \circ \iota$ as defined above are such that $\pi \circ \overline{\alpha_1}$ is residuated and $\overline{\gamma_1} \circ \iota$ is its residual, and thus form a Galois connection between moduloids \overline{B} and \overline{A} seen as lattices.*

Proof. We first note that $\pi \circ \overline{\alpha_1}$ and $\overline{\gamma_1} \circ \iota$ are monotonic by composition of monotonic mappings. We then show that $(\overline{\gamma_1} \circ \iota) \circ (\pi \circ \overline{\alpha_1}) \geq Id_{\overline{B}}$: for all $a \in \overline{B}$, $\pi(\overline{\alpha_1}(a)) \geq \overline{\alpha_1}(a)$ because π is extensive. As $\overline{\gamma_1}$ is monotonic and is the residual of $\overline{\alpha_1}$, we have $\overline{\gamma_1} \circ \pi(\overline{\alpha_1}(a)) \geq \overline{\gamma_1}(\overline{\alpha_1}(a)) \geq a$. We finally show that $(\pi \circ \overline{\alpha_1}) \circ (\overline{\gamma_1} \circ \iota) \leq Id_{\overline{\mathcal{B}(A)}}$: as $\overline{B} \xleftarrow[\overline{\alpha_1}]{\overline{\gamma_1}} \overline{\mathcal{B}(A)}$ is a Galois connection, $\overline{\alpha_1} \circ \overline{\gamma_1} \circ \iota(x) = \overline{\alpha_1} \circ \overline{\gamma_1}(x) \leq x$ for all $x \in \overline{A}$. By applying the monotonic function π to each member

of this inequality, we get $\pi(\overline{\alpha}_1 \circ \overline{\gamma}_1(x)) \leq \pi(x)$. As $x \in \overline{A}$, $\pi(x) = x$, which allows us to conclude the proof. \square

B Proof of Theorem 6

Lemma 2 *Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction and $(\sigma^\sharp, \sigma) \in \overline{ba} \times \overline{b}$. Then, we have:*

$$\bigoplus_{\{c \in \overline{b} \mid \sigma^\sharp \leq \overline{\alpha}_1(c)\}} M_{c\sigma} \leq \bigoplus_{\{a \in \overline{ba} \mid a \leq \overline{\alpha}_1(\sigma)\}} M_{\sigma^\sharp a}^\sharp. \quad (3)$$

Proof. Let first consider the left member of the correct linear abstraction inequality.

$$\begin{aligned} (\overline{\alpha}_1 \circ M)_{\sigma^\sharp \sigma} &= \bigoplus_{c \in \overline{b}} \overline{\alpha}_1 \sigma^\sharp c \otimes M_{c\sigma} \\ &= \bigoplus_{\{c \mid \sigma^\sharp \leq \overline{\alpha}_1(c)\}} M_{c\sigma} \quad \text{by definition of } \overline{\alpha}_1 \end{aligned}$$

We note in the passing that inequality $\sigma^\sharp \leq \overline{\alpha}_1(c)$ is equivalent to the fact that the element $\sigma^\sharp \in \overline{ba}$ appears in the vector decomposition of $\overline{\alpha}_1(c)$ over the basis \overline{ba} .

We conclude the proof by developing the right member of the inequality:

$$\begin{aligned} (M^\sharp \circ \overline{\alpha}_1)_{\sigma^\sharp \sigma} &= \bigoplus_{a \in \overline{ba}} M_{\sigma^\sharp a}^\sharp \otimes \overline{\alpha}_1 a \sigma \\ &= \bigoplus_{\{a \mid a \leq \overline{\alpha}_1(\sigma)\}} M_{\sigma^\sharp a}^\sharp \end{aligned}$$

\square

Lemma 3 *Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction. Then, for all $k \geq 1$, $(M^k, (M^\sharp)^k, \overline{\alpha}_1)$ is a correct linear abstraction.*

Proof. We proceed by induction over k . The property holds at rank 1 by hypothesis. If the property holds at rank n , it is also established at rank $n+1$ by applying property at rank 1 and by preservation of the order in a dioid. \square

Lemma 4 *Let Q be a selective cost dioid. Let $(M, M^\sharp, \overline{\alpha}_1)$ be a correct linear abstraction. Then, for all σ and $k \geq 1$, such that $M_{\sigma\sigma}^k \neq \perp$ and for all $\sigma_i^\sharp \in \overline{ba}$ appearing in the vector decomposition of $\overline{\alpha}_1(\sigma)$ on the basis \overline{ba} ($\Leftrightarrow \sigma_i^\sharp \leq \overline{\alpha}_1(\sigma)$), there exists $\sigma_j^\sharp \leq \overline{\alpha}_1(\sigma)$ such that $M_{\sigma_i^\sharp \sigma_j^\sharp}^{\sharp k} \geq M_{\sigma\sigma}^k$.*

Proof. Let $\sigma \in \overline{b}$ such that $M_{\sigma\sigma}^k \neq \perp$. We note $\overline{\alpha}_1(\sigma) = \sigma_1^\sharp \oplus \dots \oplus \sigma_s^\sharp$ the vector decomposition of $\overline{\alpha}_1(\sigma)$ on the basis \overline{ba} .

We apply inequality (2) to $(\sigma_1^\sharp, \sigma), \dots, (\sigma_s^\sharp, \sigma), M^k$ and $M^{\sharp k}$.

As for all i , σ belongs to $\{c \mid \sigma_i^\sharp \leq \overline{\alpha}_1(c)\}$, we get:

$$\begin{aligned} M_{\sigma\sigma}^k &\leq M_{\sigma_1^\sharp \sigma_1^\sharp}^{\sharp k} \oplus \dots \oplus M_{\sigma_s^\sharp \sigma_s^\sharp}^{\sharp k} = M_{\sigma_1^\sharp \sigma_{m_1}^\sharp}^{\sharp k} \\ &\quad \vdots \quad \vdots \\ M_{\sigma\sigma}^k &\leq M_{\sigma_s^\sharp \sigma_1^\sharp}^{\sharp k} \oplus \dots \oplus M_{\sigma_s^\sharp \sigma_s^\sharp}^{\sharp k} = M_{\sigma_s^\sharp \sigma_{m_s}^\sharp}^{\sharp k} \end{aligned}$$

where m_i denotes the index of the greatest element of the right member of the inequality (recall that we demand the dioid to be selective). Thus, for all σ_i^\sharp , $M_{\sigma_i^\sharp \sigma_{m_i}^\sharp}^{\sharp k} \geq M_{\sigma\sigma}^k$. \square

Lemma 5 Let Q be a selective cost dioid. Let $(M, M^\sharp, \overline{\alpha_1})$ be a correct linear abstraction. Let $\sigma \in \overline{b}$ and $k \geq 1$ such that $M_{\sigma\sigma}^k \neq \perp$. We note $\overline{\alpha_1}(\sigma) = \sigma_1^\sharp \oplus \dots \oplus \sigma_s^\sharp$. Then, there exist $1 \leq j, r \leq s$ such that:

$$\sqrt[k]{(M^k)_{\sigma\sigma}} \leq \sqrt[kr]{(M^{\sharp kr})_{\sigma_j^\sharp \sigma_j^\sharp}}.$$

Proof. Applying Lemma 3, there exist $(m_i)_{1 \leq i \leq s}$, elements of $\llbracket 1, s \rrbracket$ such that, for all i , $M_{\sigma_i^\sharp \sigma_{m_i}^\sharp}^{\sharp k} \geq M_{\sigma\sigma}^k$. It implies that every edge $(\sigma_i^\sharp, \sigma_{m_i}^\sharp)$ of the graph $M^{\sharp k}$ has a non-zero cost ($\neq \perp$). Every vertex of the graph $M^{\sharp k}$ restricted to the vertices $\{\sigma_1^\sharp, \dots, \sigma_s^\sharp\}$ has at least one leaving edge. We deduce from this that there exists a cycle in this restricted graph. Thus, there is $1 \leq j \leq s$ such that $M_{\sigma_j^\sharp \sigma_{l_1}^\sharp}^{\sharp k} \geq M_{\sigma\sigma}^k, \dots, M_{\sigma_{l_r}^\sharp \sigma_j^\sharp}^{\sharp k} \geq M_{\sigma\sigma}^k$ for an appropriate $r \in \llbracket 1, s \rrbracket$. By order preservation, we get:

$$aux = M_{\sigma_j^\sharp \sigma_{l_1}^\sharp}^{\sharp k} \otimes \dots \otimes M_{\sigma_{l_r}^\sharp \sigma_j^\sharp}^{\sharp k} \geq (M_{\sigma\sigma}^k)^r.$$

By definition of the diagonal elements of $M^{\sharp kr}$, we get that $M_{\sigma_j^\sharp \sigma_j^\sharp}^{\sharp kr} \geq aux \geq (M_{\sigma\sigma}^k)^r$. We recall that the kr th power is a monotonic function. Thus, it suffices to apply it to each side of the inequality to get the wanted result. \square

Now, we can establish Theorem 6.

Proof. By applying Lemma 4, we get that for each cycle c of M there exists a cycle c^\sharp of M^\sharp of higher average cost ($\tilde{q}(c^\sharp) \geq \tilde{q}(c)$). Thus,

$$\rho(M) = \bigoplus_{c \text{ cycle of } M} \tilde{q}(c) \leq \bigoplus_{c^\sharp \text{ cycle of } M^\sharp} \tilde{q}(c^\sharp) = \rho(M^\sharp).$$

\square